Métodos para la construcción de software fiable:

Abstracción y verificación de software

María del Mar Gallardo Melgarejo

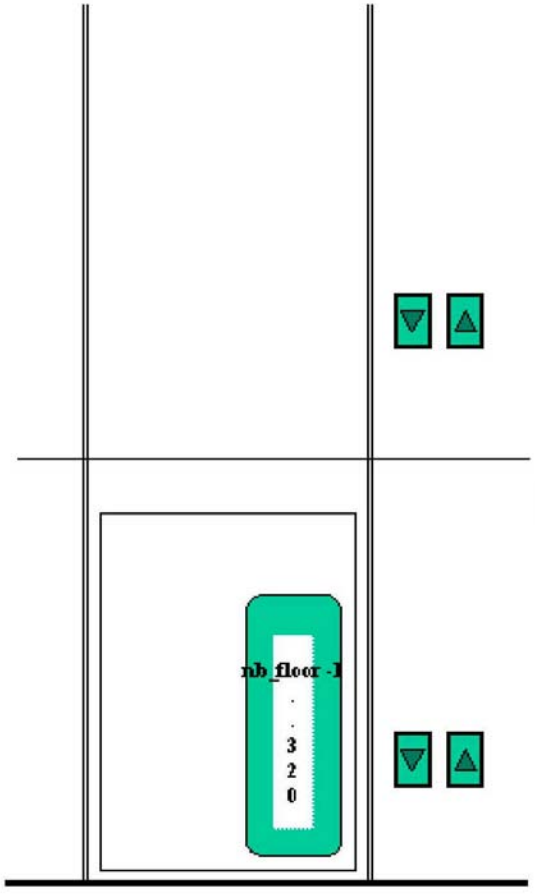
Pedro Merino Gómez

Dpto. de Lenguajes y Ciencias de la Computación

Universidad de Málaga

(gallardo,pedro)@lcc.uma.es

Ejemplo: Un ascensor



SysLift ()

SysPanel()

SysStop ()

Control part

Lift ()

Sampler ()

Scheduling

init ()

```
Position[ ]
Internal_request[ ]
```

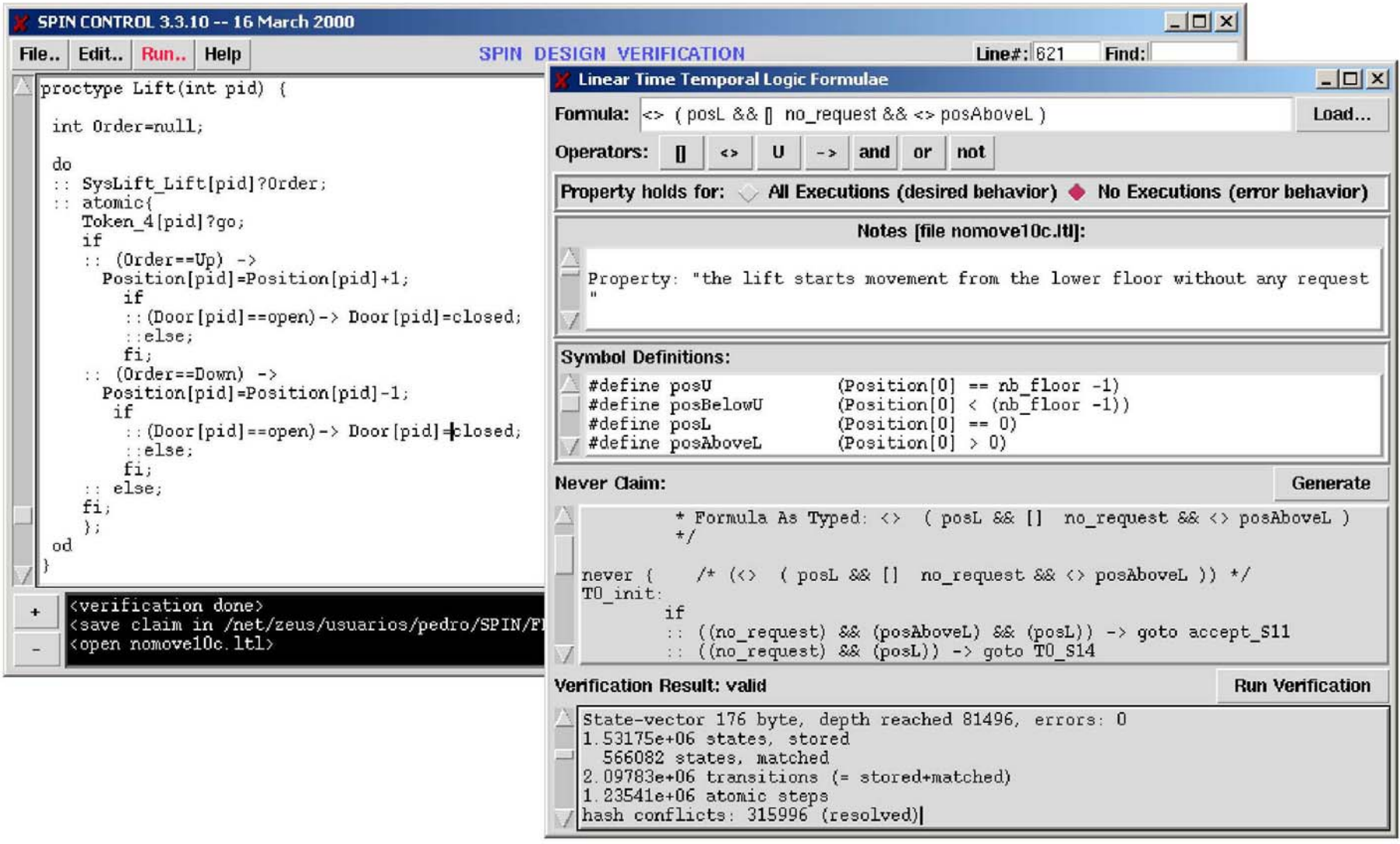
Global variables

Modelo Promela del Ascensor

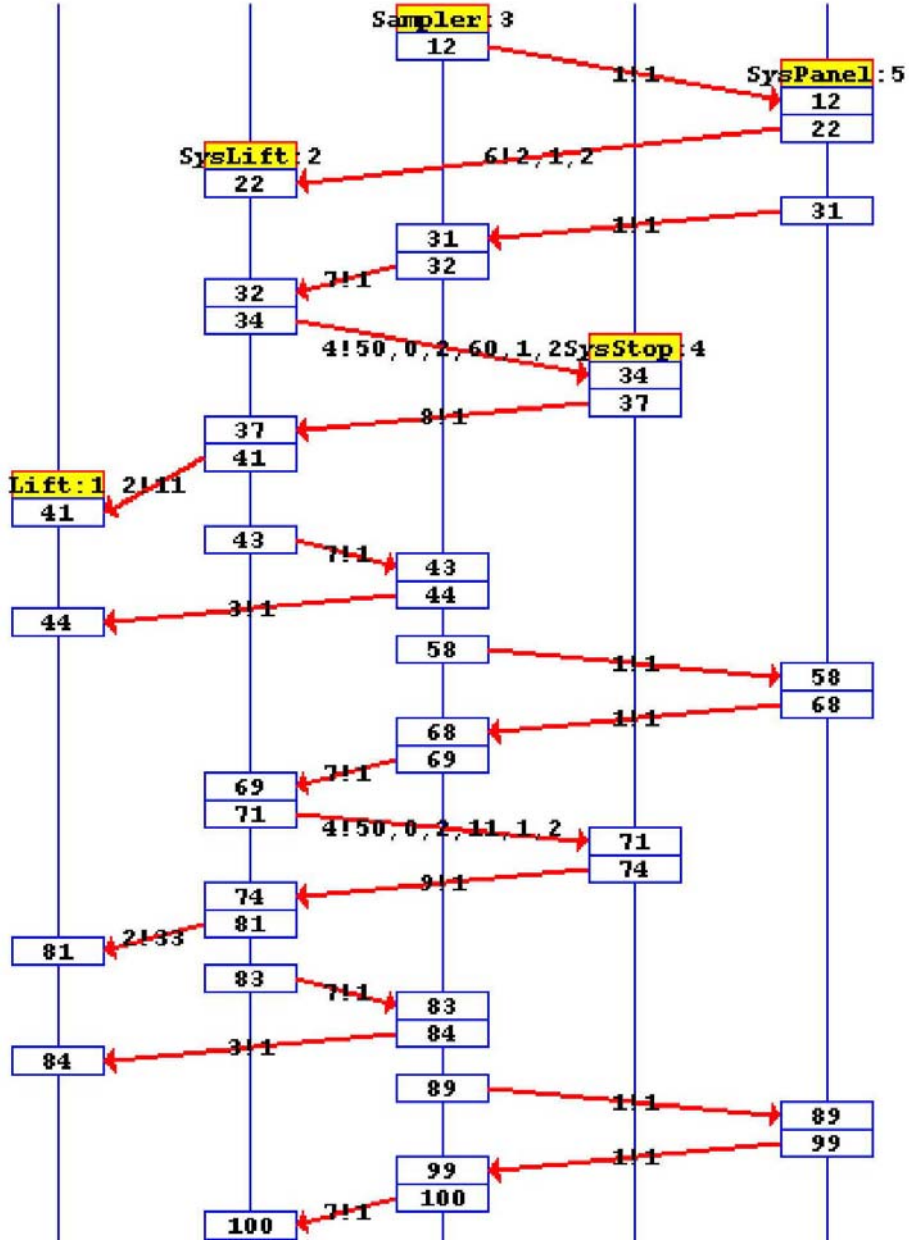
```
proctype Sampler() {

   do   /* Make one internal request and then schedule */

    ::   f = Position[0] ->
     if
     :: (f == (nb_floor - 1)) -> f = 0
     :: else -> f++
     fi;
     do   /* Choose the floor to request */
     :: ((f < (nb_floor - 1)) && (f != Position[0])) ->
        f++
     :: break
     od;


  progress_request:       internal_request[f]=true;

      /* Gives control to other processes */
      Token_1!go;
      Token_1?go;
      Token_3[0]!go;
      Token_3[0]?go;
      Token_4[0]!go;
   od }
```

```
init{
     atomic{
     if  /*  decide the initial floor */
     :: Position[0] = 0
     :: Position[0] = 1
     :: Position[0] = 2
     :: Position[0] = 3
     fi;
       /* configure the initial system */
     run Lift(0);
     run SysLift(0);
     run Sampler();
     run SysStop();
     run SysPanel();
    #ifdef CONFIG
      config = true; /* employed to check NoMove */
    #endif
     }
}
```

El modelo Promela en SPIN



Un escenario



Verificación del Modelo

Una propiedad crítica:"el ascensor no se mueve de un extremo a otro, si no hay llamadas"

No existe ninguna traza en el modelo que satisfata NoMove

NoMove: <> (configured &&
<> ((posL && [] no_request && <> posU)
|| (posU && [] no_request && <> posL)))

```
#define configured     (config==true)
#define posU           (Position[0] == (nb_floor -1))
#define posL           (Position[0] == 0)
#define no_request     ((internal_request[0] != true) &&
                        (internal_request[1] != true) &&
                        (internal_request[2] != true) &&
                        (internal_request[3] != true))
```

Verificación del Modelo

```
#define configured     (config==true)
#define posU           (Position[0] == (nb_floor -1))
#define posL           (Position[0] == 0)

#define no_request     ((internal_request[0] != true) &&
                        (internal_request[1] != true) &&
                        (internal_request[2] != true) &&
                        (internal_request[3] != true)) .....
```

La verificación depende del número de pisos

Verificación del Modelo

	Floors	3	4	5	6	7	8	9	10
Modelo concreto	Safety	2772	7942	21582	57467	150692	388440	982694	2.4392e+06
	Move	3184	9164	24974	66436	173566	445179	1.12038e+06	2.76741e+06
	NoMove	5620	15983	43286	115079	301552	777071	1.9656e+06	4.87864e+06

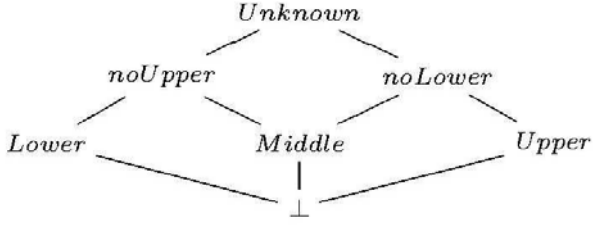
Abstracción

Paso 1. Abstracción de datos

$$\beta : [0..(nb\_floor - 1)] \mapsto \texttt{FLOORS}$$

$$\beta(0) = Lower$$
$$\beta(nb\_floor - 1) = Upper$$
$$\beta(j) = Middle, \forall 0 < j < nb\_floor - 1$$

	inc1	dec1
$Lower$	$Middle$	$\bot$
$Middle$	$noLower$	$noUpper$
$Upper$	$\bot$	$Middle$
$noLower$	$noLower$	$noUpper$
$noUpper$	$noLower$	$noUpper$
$Unknown$	$noLower$	$noUpper$



Operaciones abstractas

Retículo Floors

Abstracción

Paso 2. Transformación del Modelo

```
#define FLR_INCR(x) if
                :: x==Lower -> x = Middle \
                :: x==Middle ->x = noLower \
                :: x==noUpper ->x = noLower \
                :: x==noLower ->x = noLower \
                :: x==Unknown ->x = noLower \
                :: else -> x = ILLEGAL \
                fi
```

```
proctype Lift(int pid){ int Order=null;
  do
  ...
  :: SysLift_Lift[pid]?Order;
     if
     :: (Order==Up) -> FLR_INCR(Position[pid]);
  ...
}
```

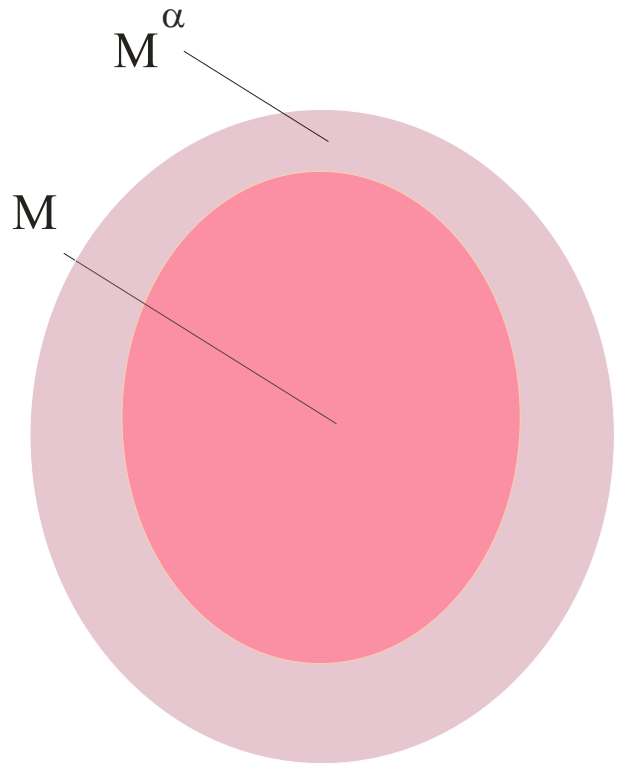
```
#define FLR_EQunder(x,y)  ((x==Lower && y==Lower) ||
                    (x==Upper && y==Upper) )
#define FLR_EQimp(x,y) (
   ((x==Upper)&&(y==noLower)) ||((x==noLower)&&(y==Upper)) ||
   ((x==Lower)&&(y==noUpper)) ||((x==noUpper)&&(y==Lower)) ||
   ((x==Middle)&&(y==noUpper))||((x==noUpper)&&(y==Middle))||
   ((x==Middle)&&(y==noLower))||((x==noLower)&&(y==Middle))||
   ((x==Middle)&&(y==Middle))||((x==Unknown)) || ((y==Unknown)))

#define FLR_EQ(x,y) (FLR_EQimp(x,y) || FLR_EQunder(x,y))
```

Corrección de la abstracción

Dado un modelo M, construir una sobreaproximación M^{α}

M y M^{α} deben satisfacer
alguna condición de corrección

Una posibilidad es que
M^{α} simule a M
aunque hay otras opciones



Corrección

Dada la abstracción $\alpha: \Sigma \rightarrow \Sigma^{\alpha}$

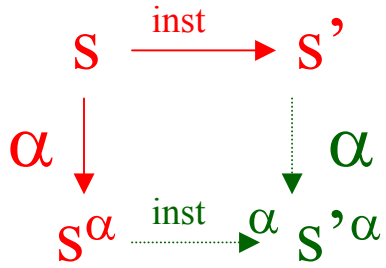
$M = (\Sigma, Inst, s_0, \rightarrow)$ $\quad\quad$ $M^{\alpha} = (\Sigma^{\alpha}, Inst, s_0^{\alpha}, \rightarrow^{\alpha})$

M^{α} es una α–simulación de M sii

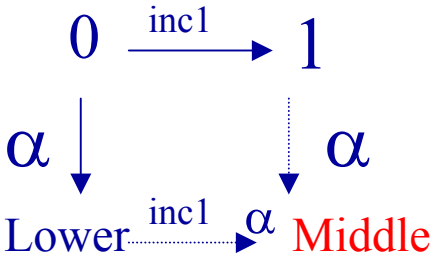
$\forall s,s' \in \Sigma, \forall inst \in Inst, \forall s^{\alpha} \in \Sigma^{\alpha}$

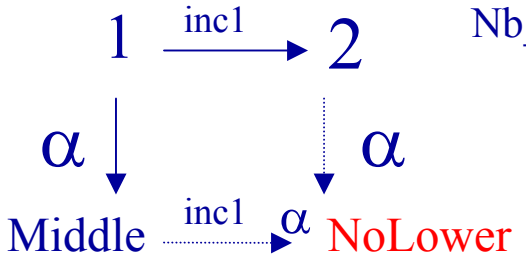
$\alpha(s) \leq s^{\alpha}, s \xrightarrow{inst} s' \Rightarrow \exists\, s'^{\alpha} \in \Sigma^{\alpha} \mid \alpha(s') \leq s'^{\alpha}, s^{\alpha} \xrightarrow{inst}{}^{\alpha} s'^{\alpha}$

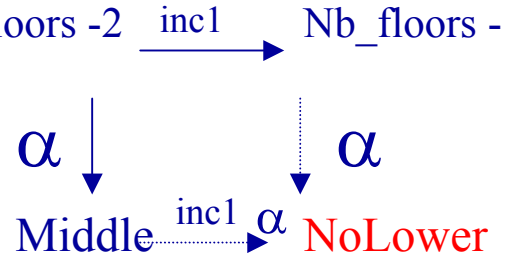
Corrección



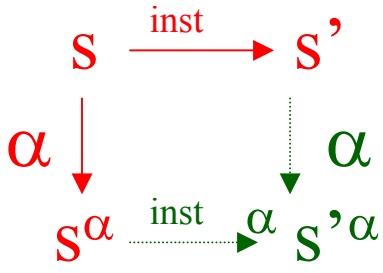
Instrucciones



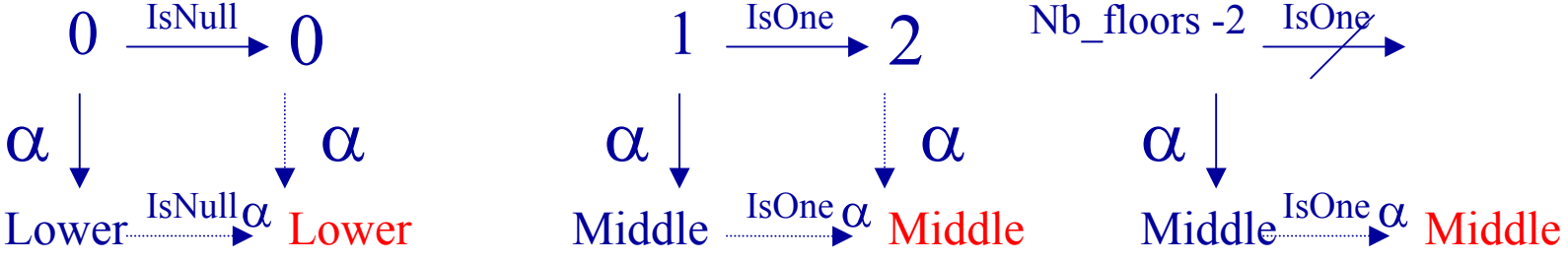




Corrección



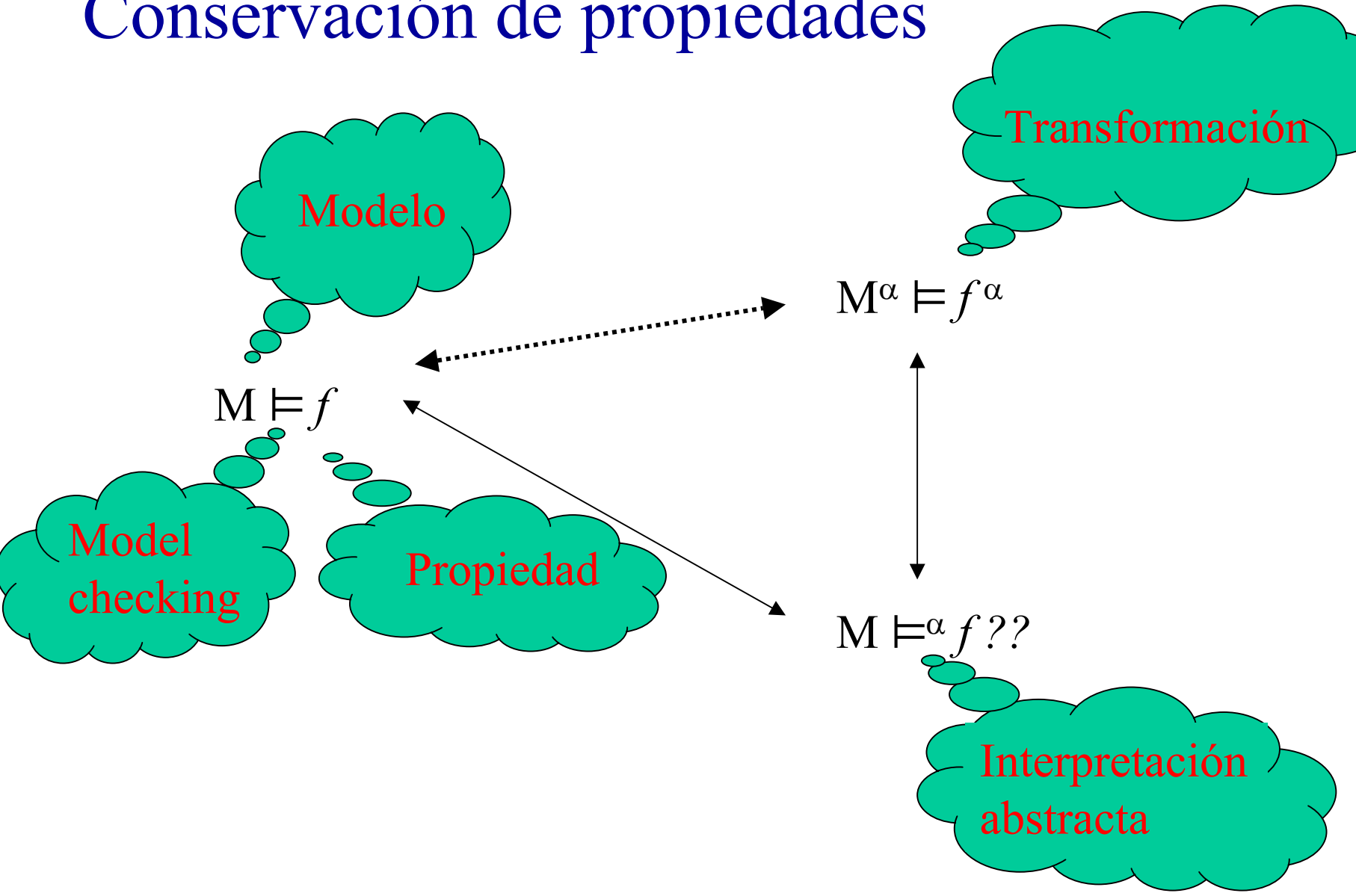
tests



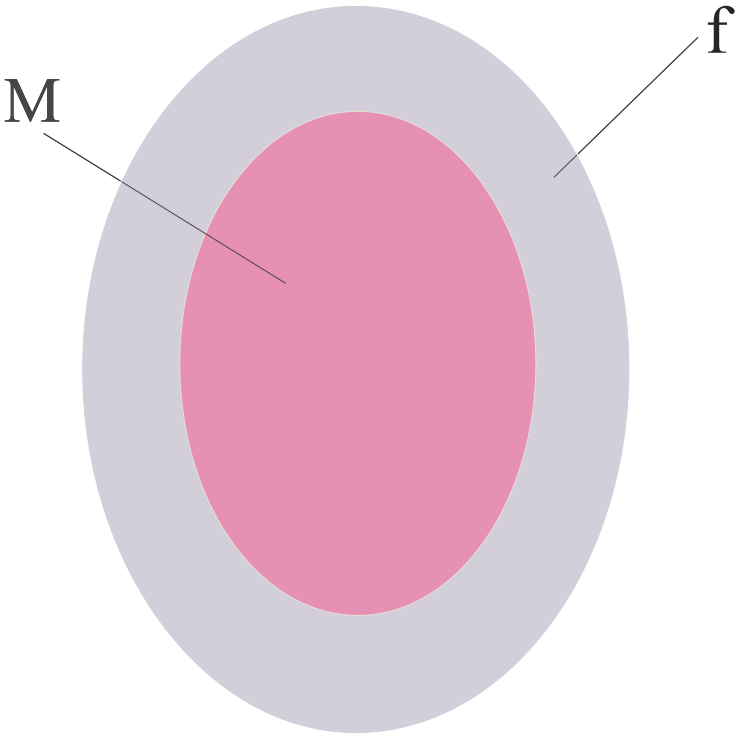
$$test(p,s) \Rightarrow test_o^{\alpha}(p,s^{\alpha})$$

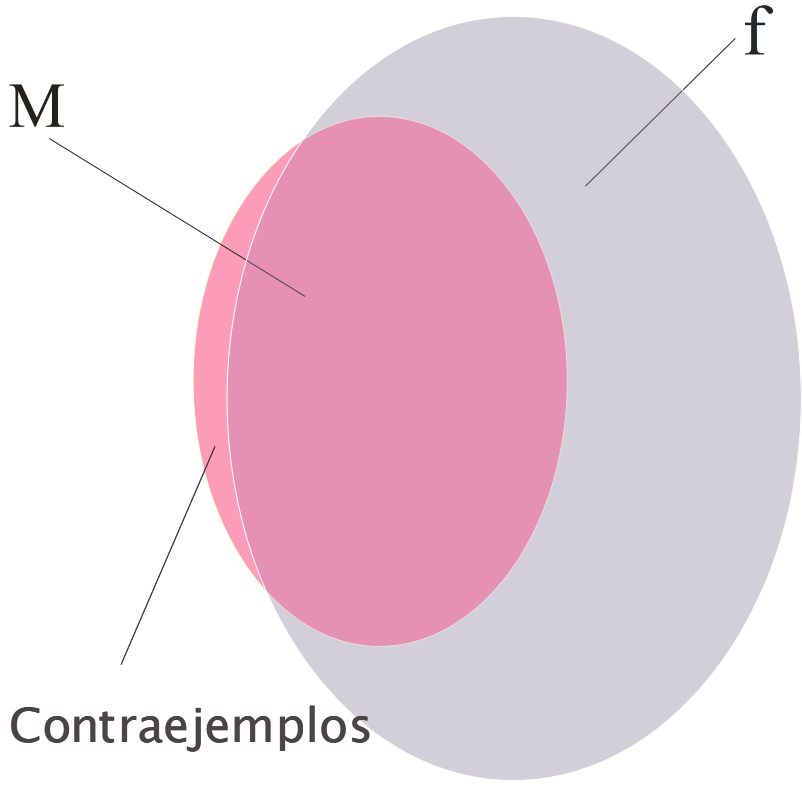
$$test_o^{\alpha}(b,s^{\alpha}) = \bigvee_{\{s|\alpha(s) \leq s^{\alpha}\}} test(p,s)$$

Conservación de propiedades

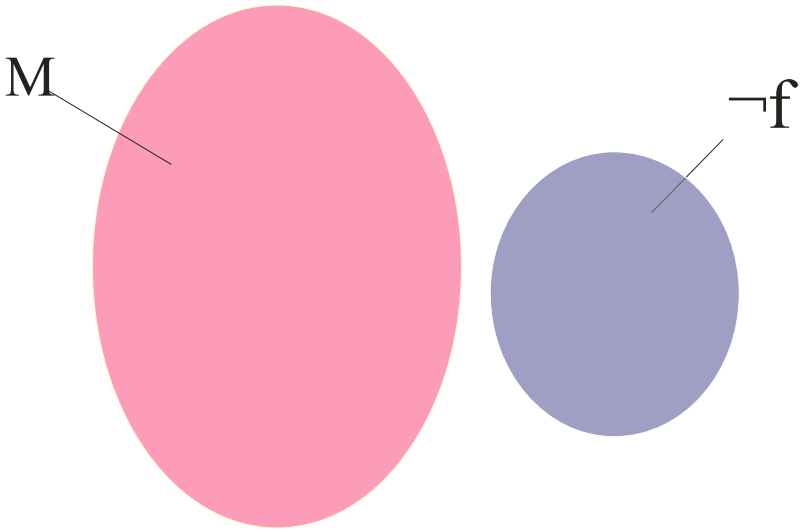


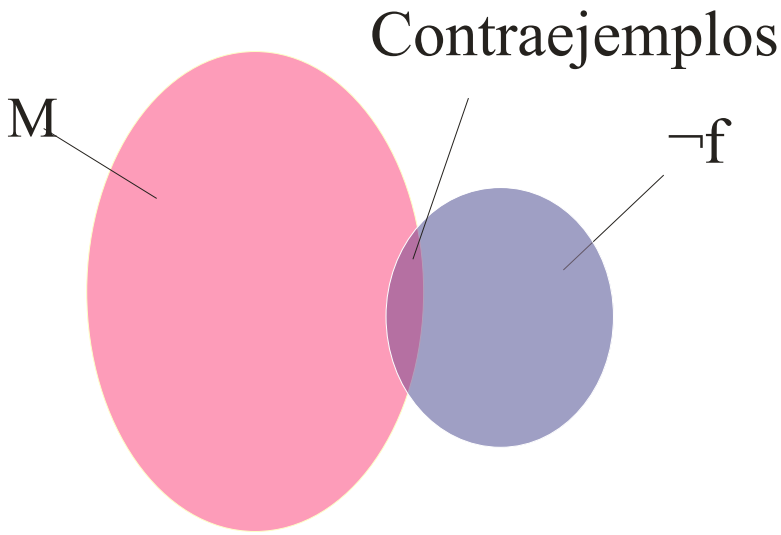
Model checking





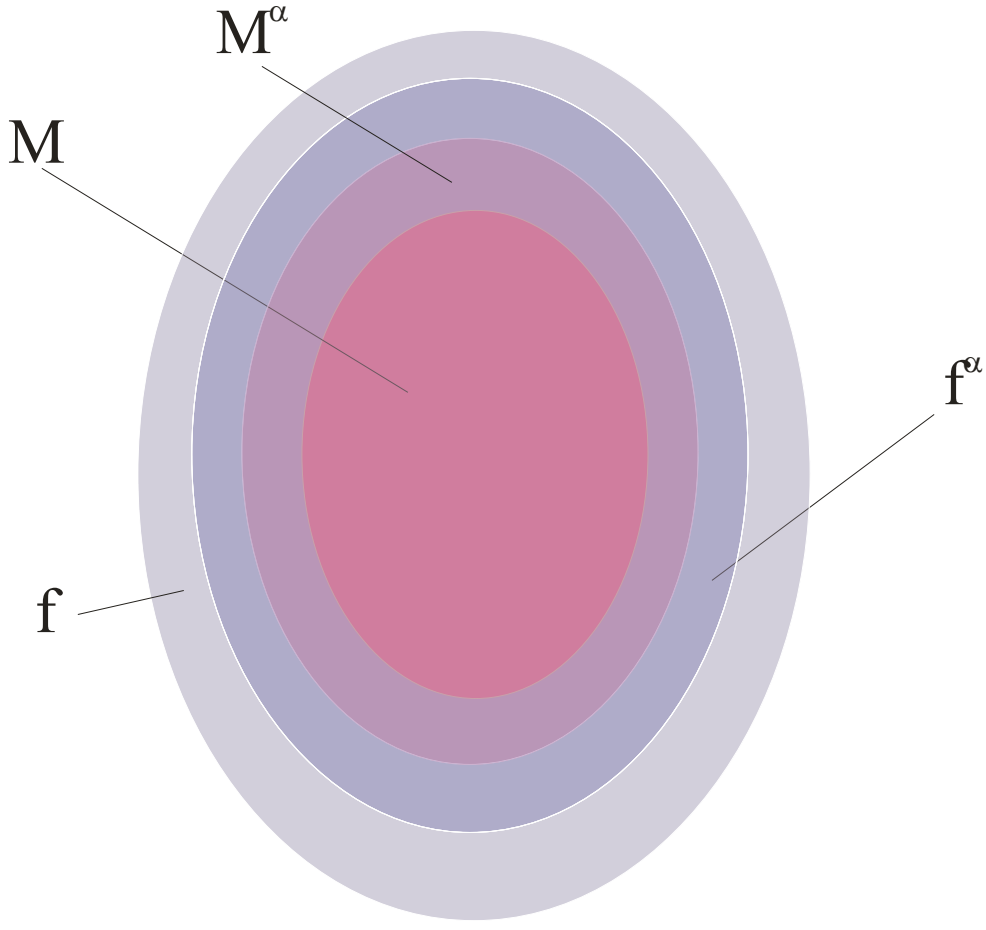
Model checking





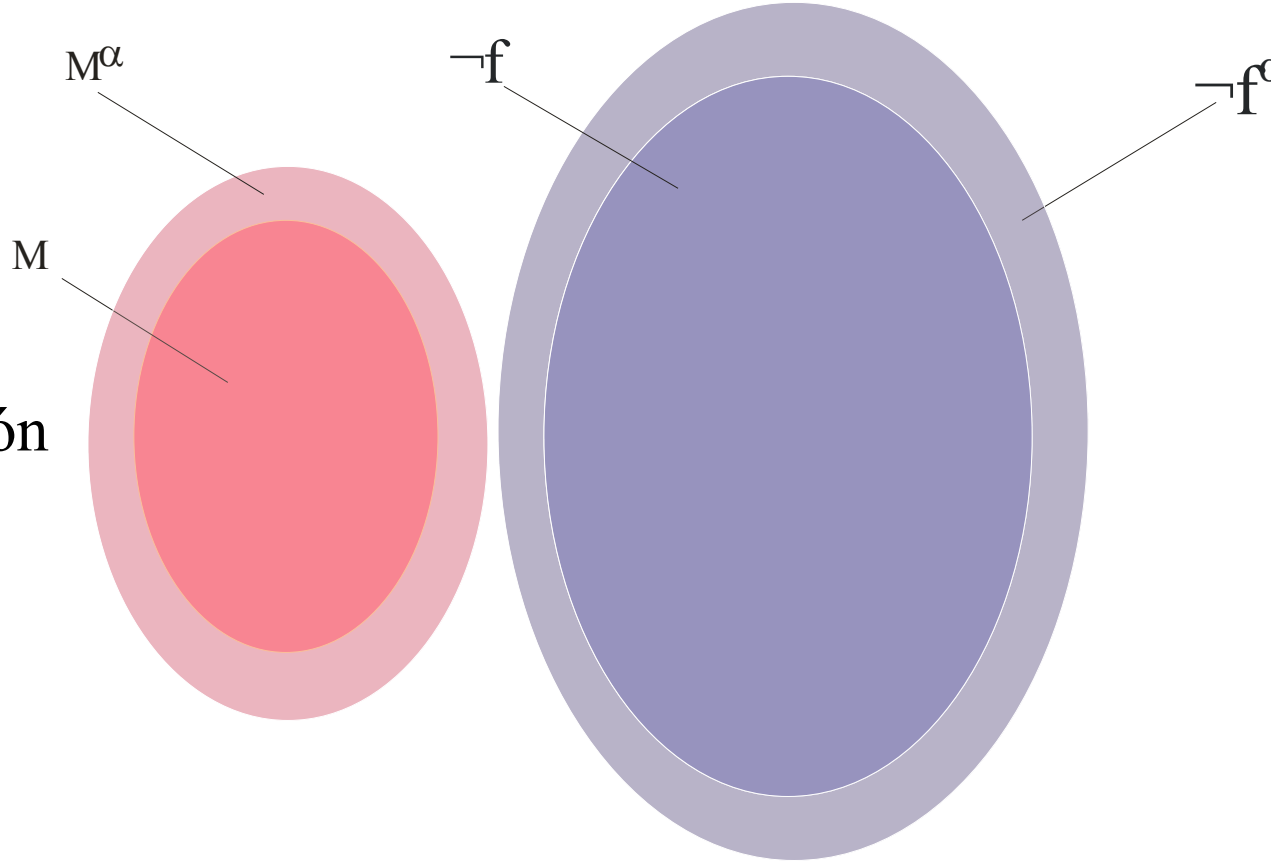
Model checking + abstracción

Subaproximación de propiedades



Model checking + abstracción

Sobreaproximación de propiedades



Semántica LTL

$$
\begin{array}{lcl}
t_i \models p & \Longleftrightarrow & s_i \models p \quad (test(p,s)) \\
t_i \models \Box f & \Longleftrightarrow & \forall j \geq i, t_j \models f \\
t_i \models \Diamond f & \Longleftrightarrow & \exists j \geq i \mid t_j \models f \\
t_i \models Of & \Longleftrightarrow & t_{i+1} \models f \\
t_i \models fUg & \Longleftrightarrow & \exists k \geq i \mid t_k \models g \ y \\
 & & \forall\, i \leq j < k, t_j \models f \\
M \models \forall f & \Longleftrightarrow & \forall \ traza\ t, t \models f \\
M \models \exists f & \Longleftrightarrow & \exists una\ traza\ t \mid t \models f
\end{array}
$$

$t_i = s_i, s_{i+1}, \ldots$

1. $s \models p \iff test(p,s)$
2. $s^\alpha \models_o^\alpha p \iff test_o^\alpha(p,s^\alpha)$
3. $s^\alpha \models_u^\alpha p \iff test_u^\alpha(p,s^\alpha)$

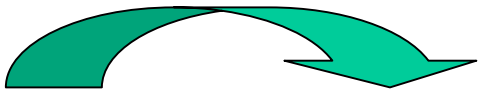
$$test_o^{\alpha}(p,s^{\alpha}) = \bigvee_{\{s \mid \alpha(s) \leq s^{\alpha}\}} test(p,s)$$

$$test_u^{\alpha}(p,s^{\alpha}) = \bigwedge_{\{s \mid \alpha(s) \leq s^{\alpha}\}} test(p,s)$$

Conservación de propiedades

1. $M^{\alpha} \models_{u}^{\alpha} \forall f \Rightarrow M \models \forall f$
2. $M^{\alpha} \not\models_{o}^{\alpha} \exists f \Rightarrow M \not\models \exists f$
3. $M^{\alpha} \models_{u}^{\alpha} \forall f \Leftrightarrow M \not\models_{o}^{\alpha} \exists f$

Transformación de propiedades



1. $s^\alpha \models^\alpha_o p \Leftrightarrow s^\alpha \models p^\alpha_o$
2. $s^\alpha \models^\alpha_u p \Leftrightarrow s^\alpha \models p^\alpha_u$

1. $t^\alpha \models^\alpha_o f \Leftrightarrow t^\alpha \models f^\alpha_o$
2. $t^\alpha \models^\alpha_u f \Leftrightarrow t^\alpha \models f^\alpha_u$

Transformación de propiedades

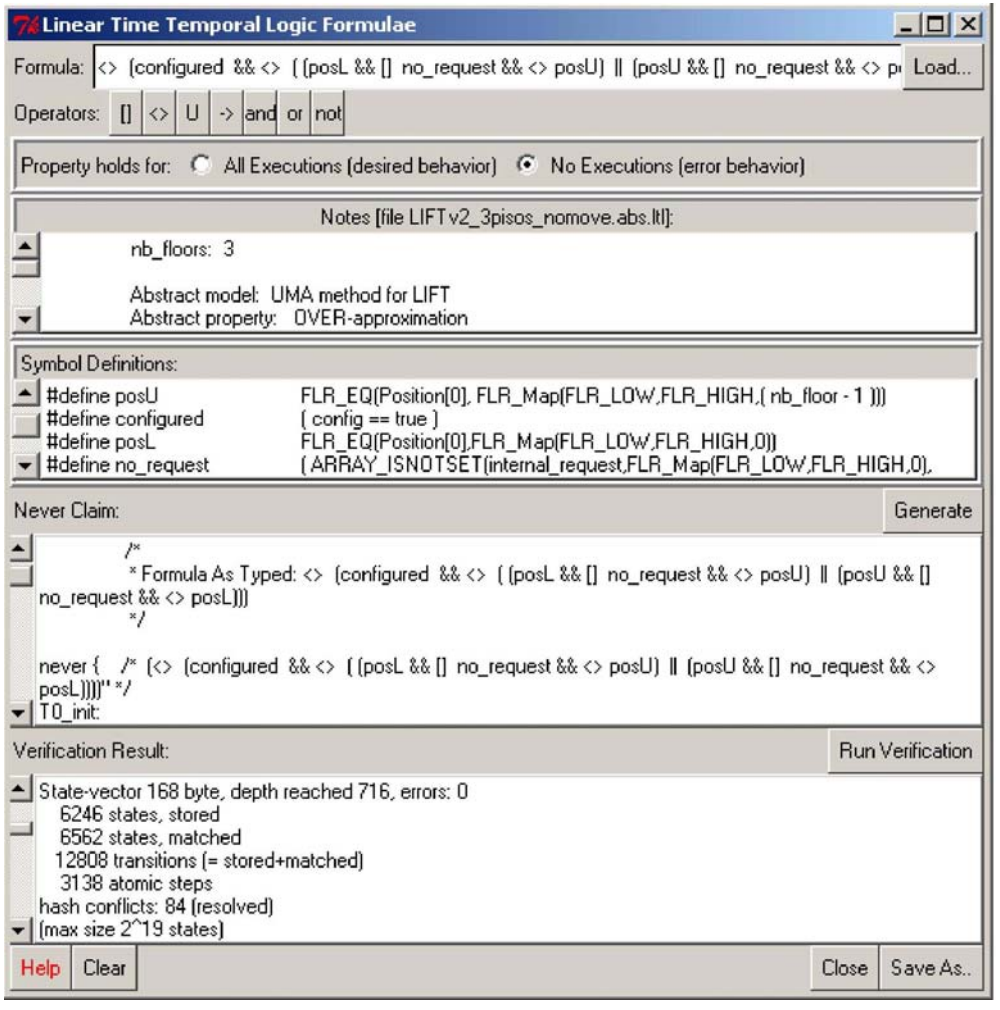
NoMove: <> (configured &&

<> ((posL && [] no_request && <> posU)

|| (posU && [] no_request && <> posL)))

```
#define configured  (config == true)
#define posU   FLR_EQ(Position[0],FLR_Map(FLR_LOW,FLR_HIGH,(nb_floor-1)))
#define posL   FLR_EQ(Position[0],FLR_Map(FLR_LOW,FLR_HIGH,0))
#define no_request (ARRAY_ISNOTSET(internal_request,
                                   FLR_Map(FLR_LOW,FLR_HIGH,0), true)
                 && ARRAY_ISNOTSET(internal_request,
                                   FLR_Map(FLR_LOW,FLR_HIGH,(nb_floor/2)), true)
                && ARRAY_ISNOTSET(internal_request,
                                   FLR_Map(FLR_LOW,FLR_HIGH,(nb_floor-1)), true)
```

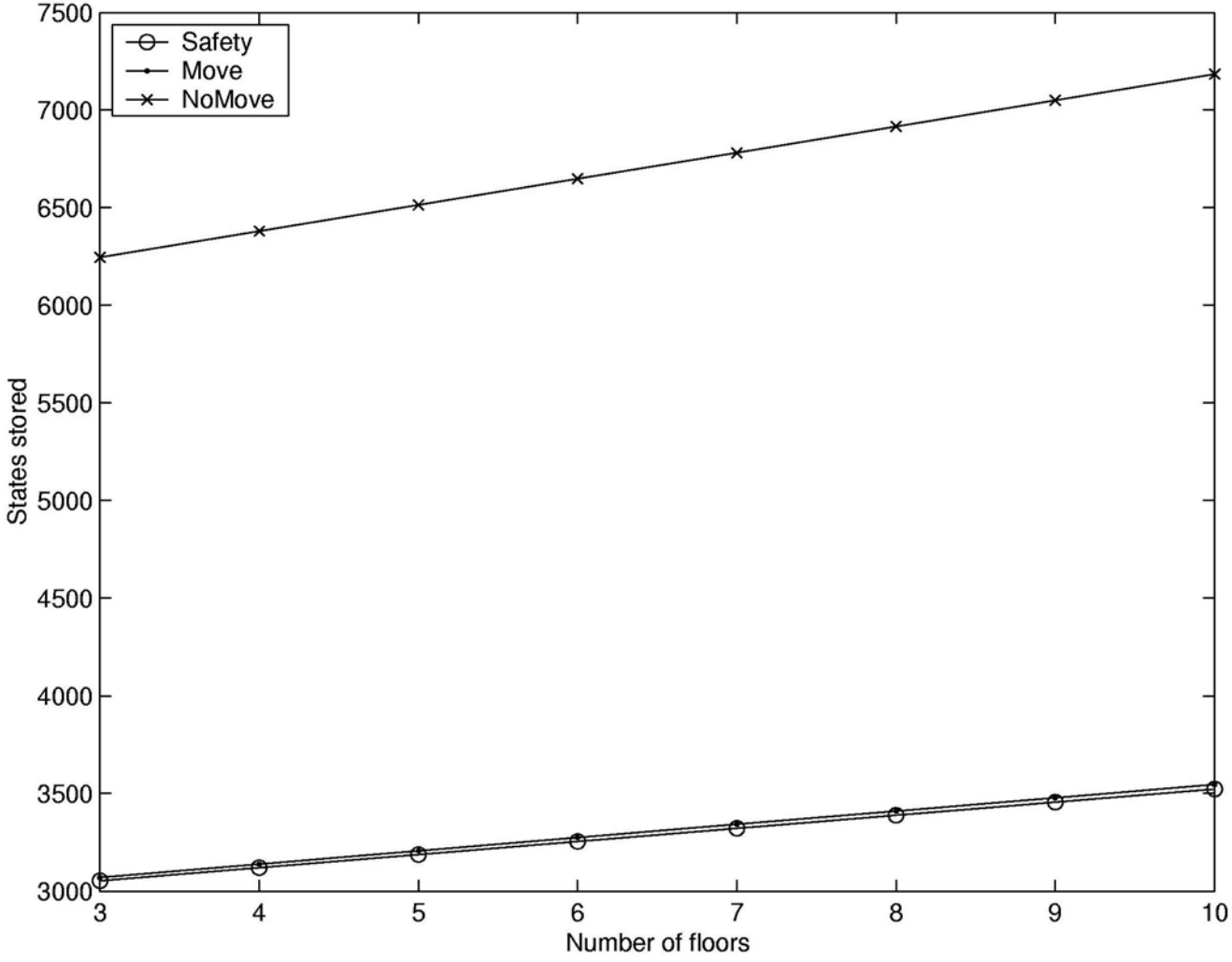
Transformación de propiedades



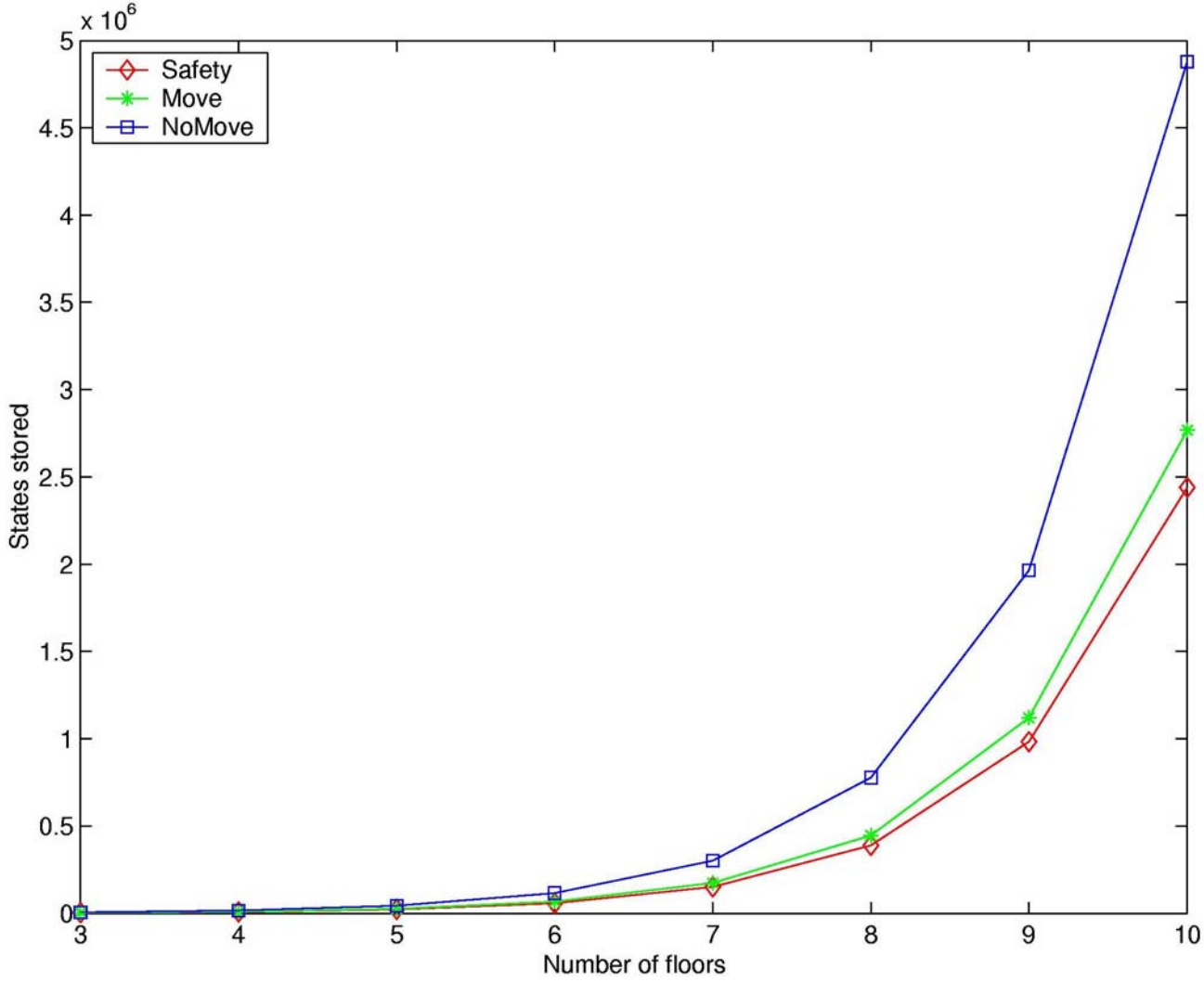
Verificación del modelo abstracto

	Floors	3	4	5	6	7	8	9	10
Modelo concreto	Safety	2772	7942	21582	57467	150692	388440	982694	2.4392e+06
	Move	3184	9164	24974	66436	173566	445179	1.12038e+06	2.76741e+06
	NoMove	5620	15983	43286	115079	301552	777071	1.9656e+06	4.87864e+06
Modelo abstracto	Safety	3053	3120	3187	3254	3321	3388	3455	3522
	Move	3070	3138	3206	3274	3342	3410	3478	3546
	NoMove	6246	6380	6514	6648	6782	6916	7050	7184

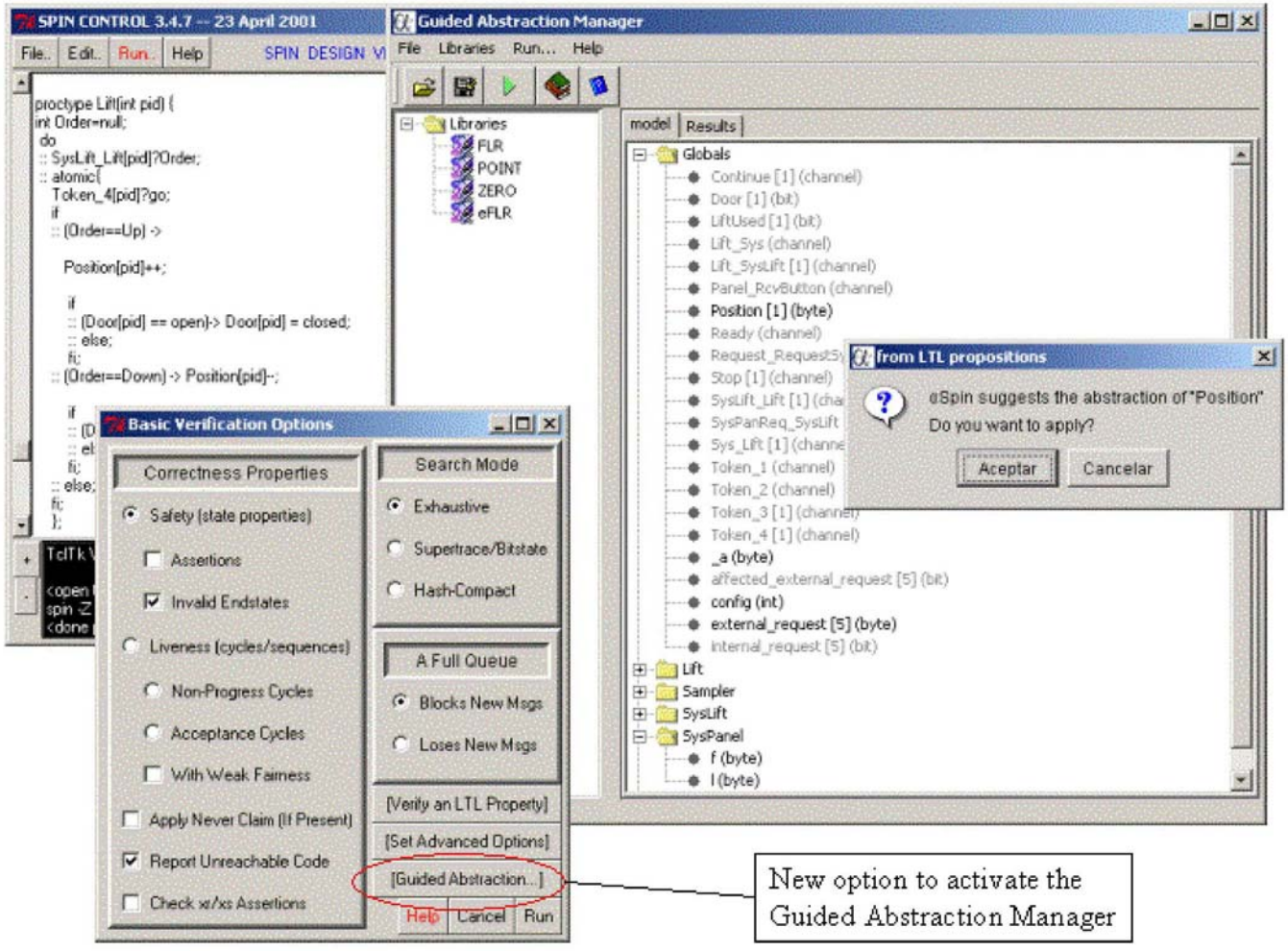
Verificación del modelo abstracto



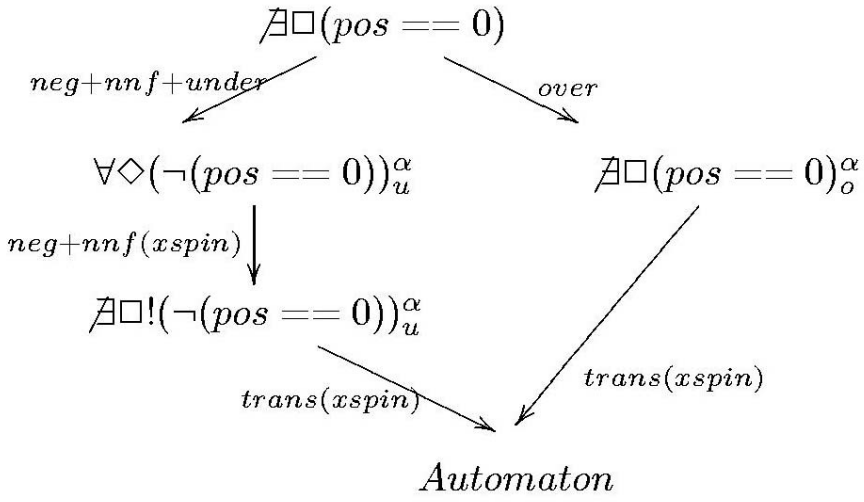
Verificación del modelo abstracto



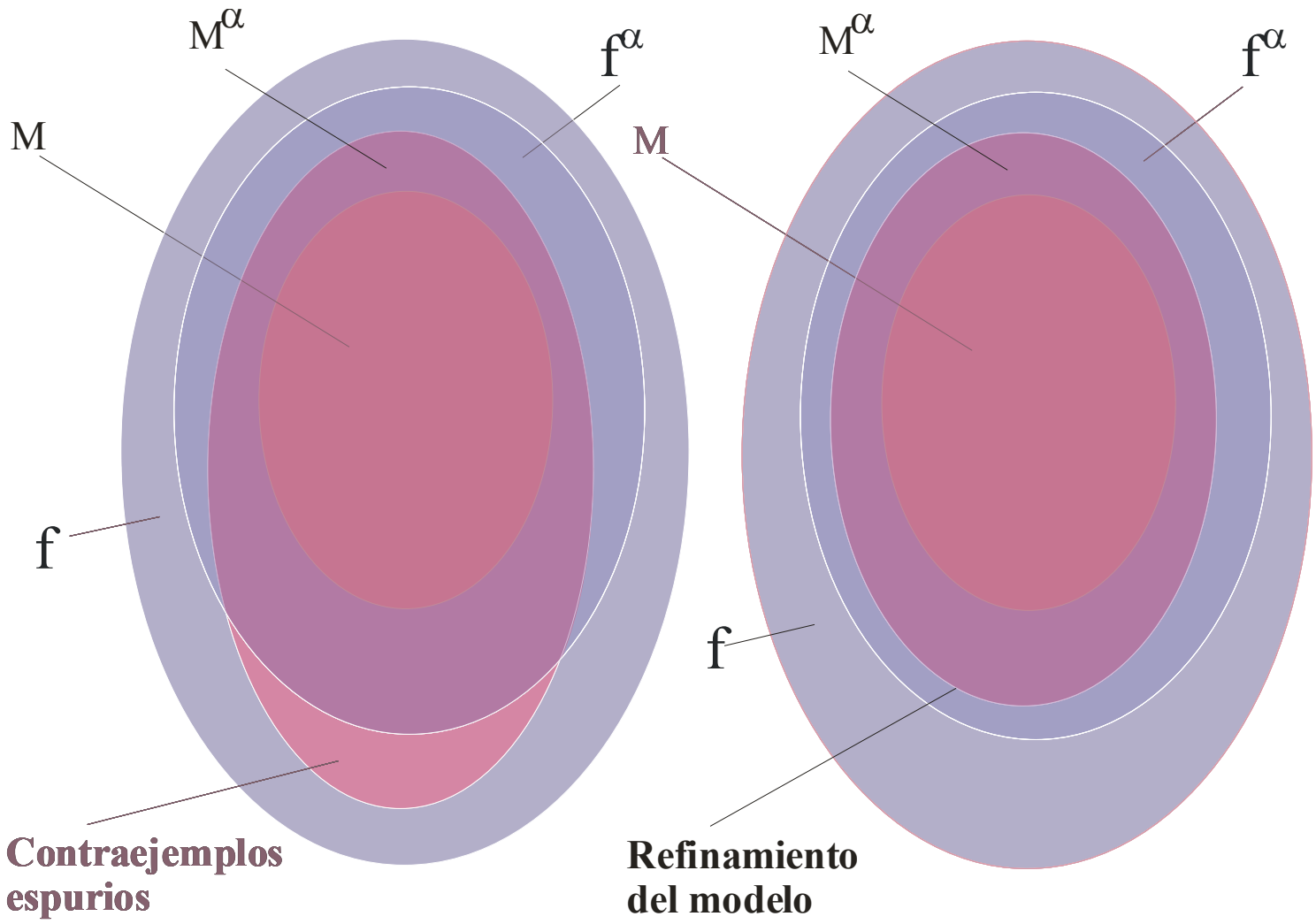
Vista de αSpin



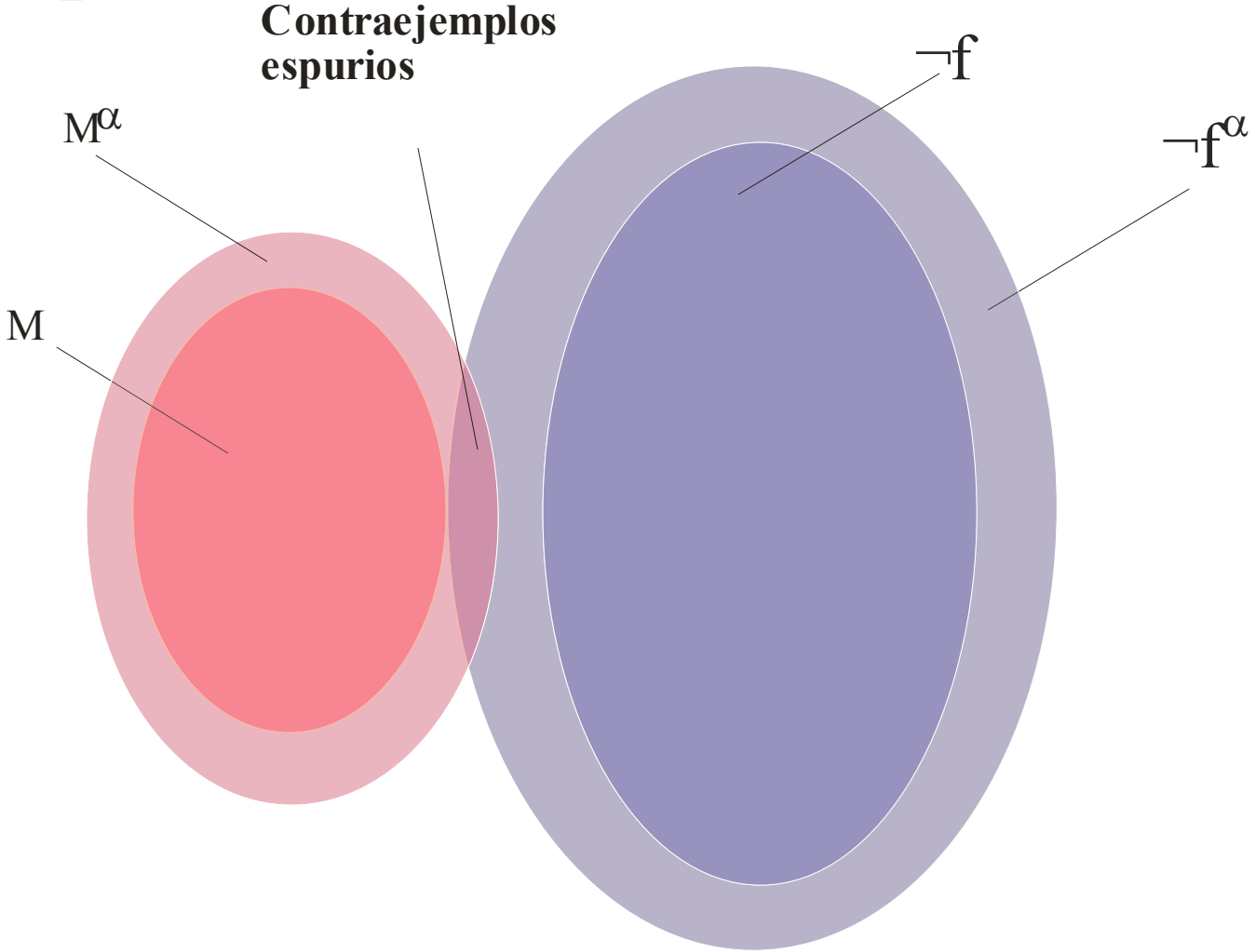
Model checking on-the-fly vs sobre y sub aproximación



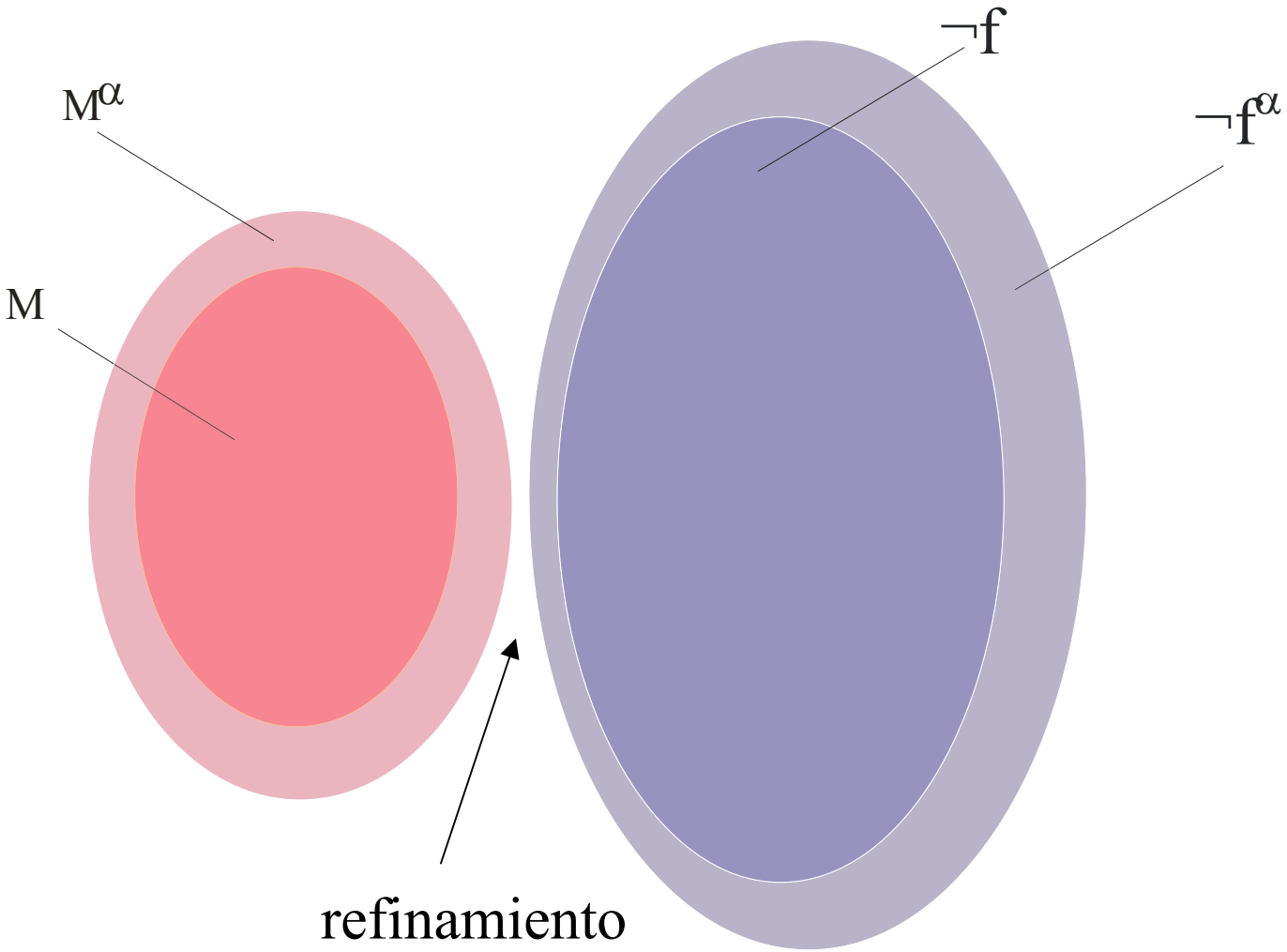
Trazas espurias



Trazas espurias



Trazas espurias



Refinamiento

1. $M \models \forall g,\ M^{\alpha} \models \forall(g_o^{\alpha} \rightarrow f_u^{\alpha}) \Rightarrow M \models \forall f$
2. $M \not\models \exists g,\ M^{\alpha} \models \forall(f_o^{\alpha} \rightarrow g_u^{\alpha}) \Rightarrow M \not\models \exists f$

Combina sobre y subaproximación de propiedades

Explota el poder del model checking para analizar la parte del modelo que nos interesa

Ejemplo

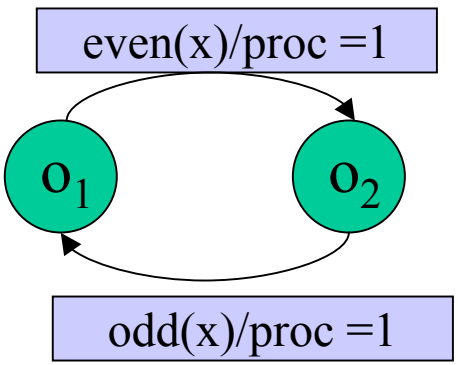
Sistema de transición

Variables globales

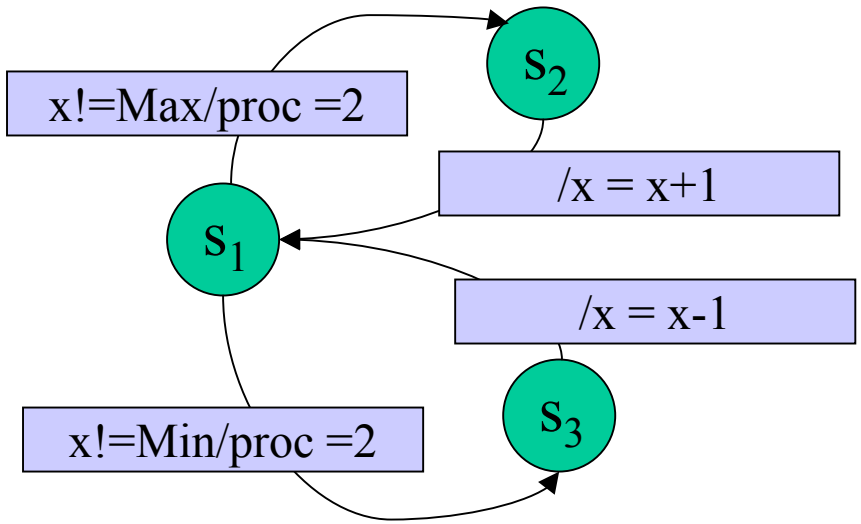
Int x, proc;

x = Min;

Process P_1



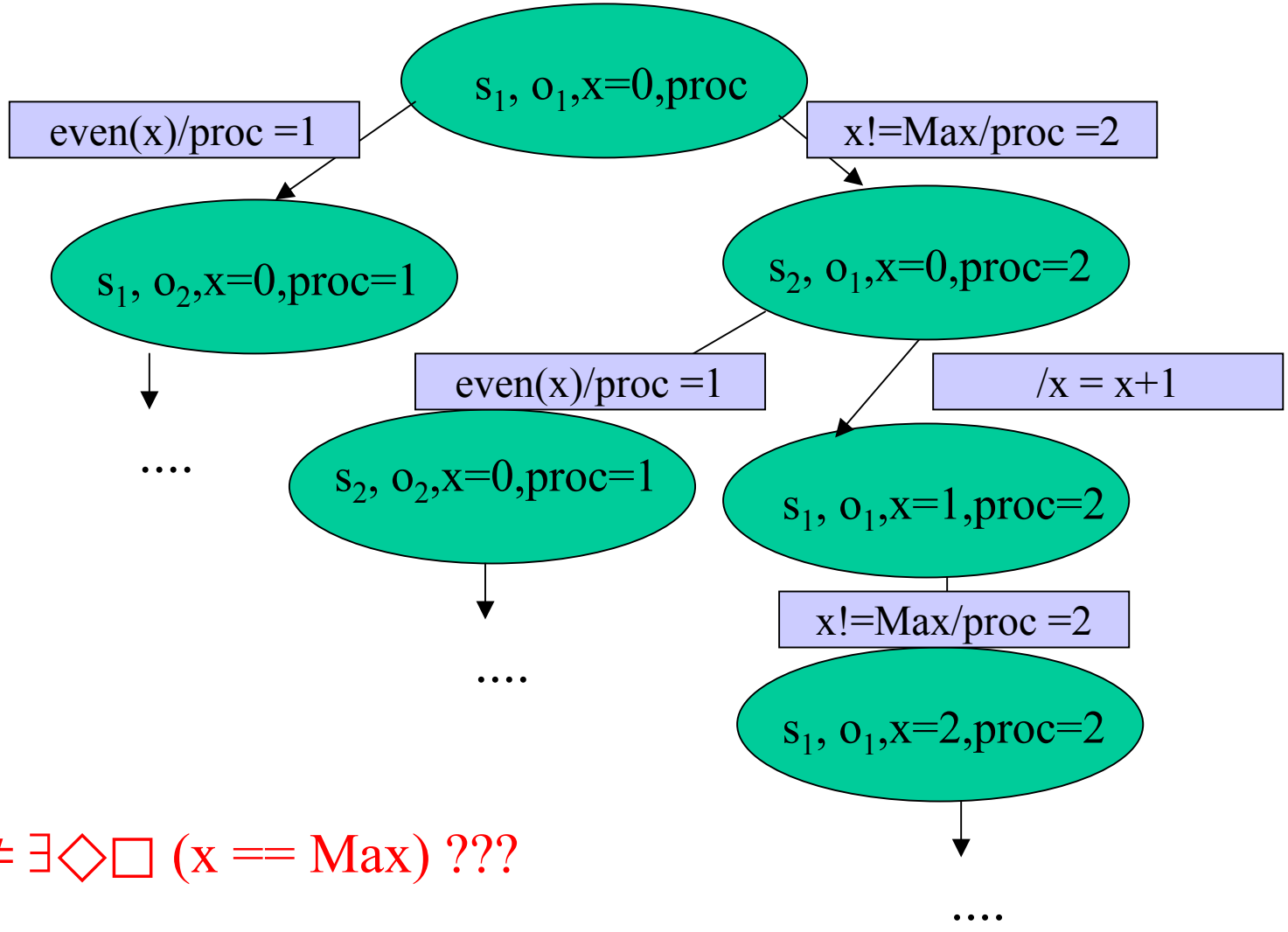
Process P_2



noprogress = $\Diamond\Box$ (proc == 1) M $\not\models \exists$ noprogress

Ejemplo

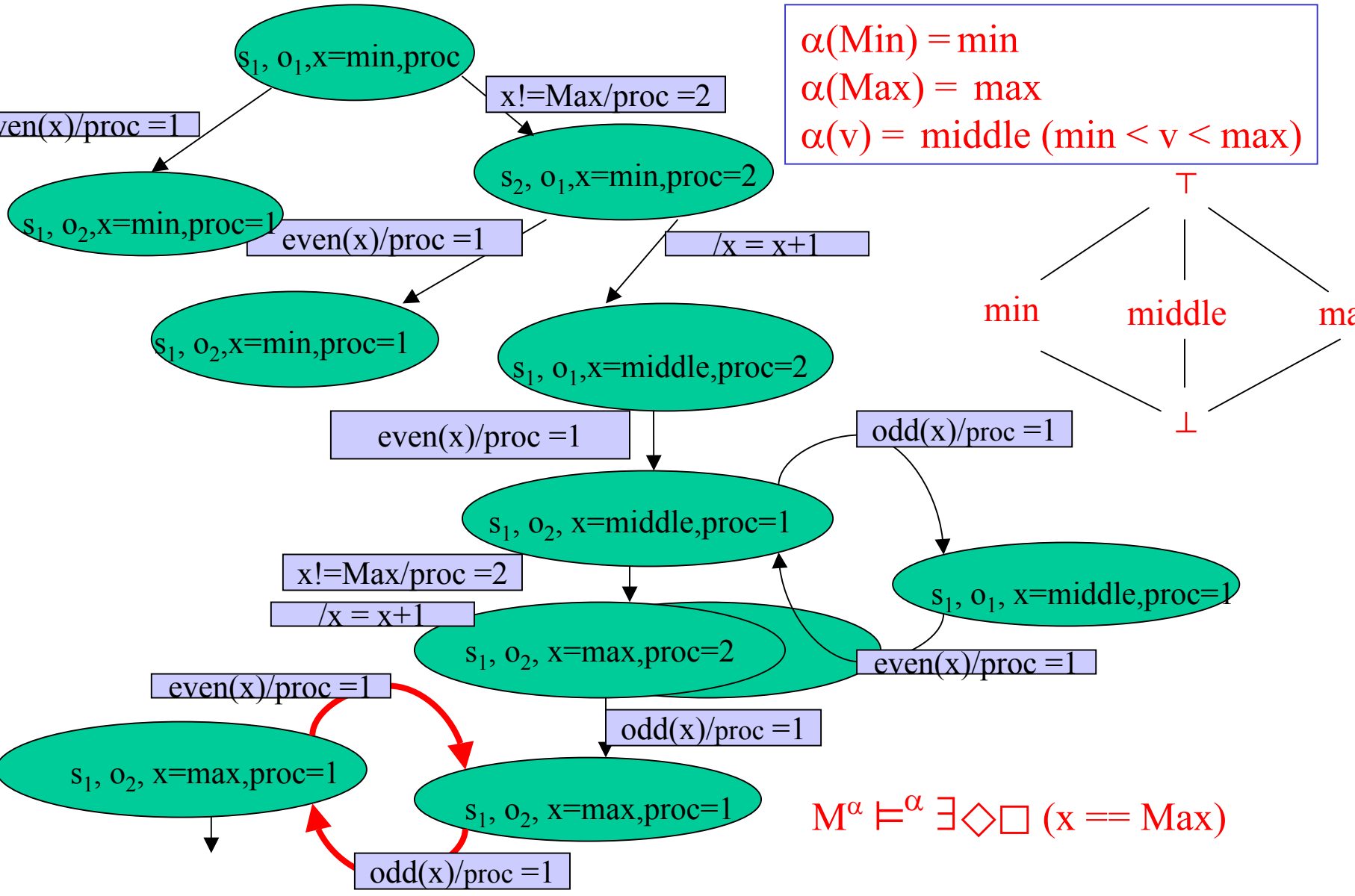
Modelo concreto



$M \not\models \exists \Diamond \Box (x == Max)$???

Ejemplo

Modelo abstracto



Ejemplo

$$M \not\models \exists \text{ noprogress}$$

$$M^{\alpha} \Vdash \forall (\Diamond\Box (x == Max)^{\alpha} \rightarrow \text{noprogress})$$

$$\Downarrow$$

$$M \not\models \exists\Diamond\Box (x == Max)$$

Hot topics en Técnicas formales

- Extensión de las técnicas de análisis para estudiar aspectos de tiempo real, rendimiento, planificabilidad,...
- Explorar distintos modelos matemáticos que permitan representar nuevos sistemas o problemas: por ejemplo, sistemas híbridos, integración de los principios matemáticos de abstracción, composición y jerarquía.
- Integración sin costuras de las nuevas herramientas desarrolladas en el proceso del desarrollo del software.
- Verificación probabilística.

Bibliografía

- Dennis Dams: Abstraction in Software Model Checking: Principles and Practice (Tutorial Overview and Bibliography). SPIN 2002: 14-21
- M. M. Gallardo, P.Merino, E.Pimentel Refinement of LTL Formulas for Abstract Model Checking The 9th International Static Analysis Symposium SAS '02 LNCS v. 2477 pp 395-410
- M. M. Gallardo, J. Martínez, P.Merino, E.Pimentel αSPIN: A Tool for Abstract Model Checking International Journal on Software Tools for Technology Transfer (en imprenta)
- J. Wing Platitudes and attitudes International Journal on Software Tools for Technology Transfer (2003) 4: 261-265